Deep Learning for Virtual Screening: Five Reasons to Use ROC Cost Functions

Vladimir Golkov [1], Alexander Becker [1], Daniel T. Plop [1], Daniel Čuturilo [1], Neda Davoudi [1], Jeffrey Mendenhall [2], Rocco Moretti [2], Jens Meiler [2,3], Daniel Cremers [1]

[1] Computer Vision Group, Technical University of Munich, Germany
[2] Center for Structural Biology, Vanderbilt University, USA
[3] Institute for Drug Discovery, Leipzig University, Germany
{vladimir.golkov, alexander.becker, daniel.plop, daniel.cuturilo, neda.davoudi, cremers}@tum.de, {jeffrey.l.mendenhall, jens.meiler}@vanderbilt.edu, rmorettiase@gmail.com

Abstract

Computer-aided drug discovery is an essential component of modern drug development. Therein, deep learning has become an important tool for rapid screening of billions of molecules *in silico* for potential hits containing desired chemical features. Despite its importance, substantial challenges persist in training these models, such as severe class imbalance, high decision thresholds, and lack of ground truth labels in some datasets. In this work we argue in favor of directly optimizing the receiver operating characteristic (ROC) in such cases, due to its robustness to class imbalance, its ability to compromise over different decision thresholds, certain freedom to influence the relative weights in this compromise, fidelity to typical benchmarking measures, and equivalence to positive/unlabeled learning. We also propose new training schemes (coherent mini-batch arrangement, and usage of out-of-batch samples) for cost functions based on the ROC, as well as a cost function based on the logAUC metric that facilitates early enrichment (i.e. improves performance at high decision thresholds, as often desired when synthesizing predicted hit compounds). We demonstrate that these approaches outperform standard deep learning approaches on a series of PubChem high-throughput screening datasets that represent realistic and diverse drug discovery campaigns on major drug target families.

1 Introduction

Drug discovery is a long, complex, and expensive process, through which new chemical compounds can be identified and that, if successful, may lead to new pharmaceutical drugs. In the testing process of compounds to be identified as potential drugs, we generally have many more negative samples than positive ones. A preselection *in silico* (virtual screening) using machine learning must select only a small percentage among millions of candidates, because running subsequent *in vitro* experiments on each preselected compound is expensive. In other words, machine learning for virtual screening is used with high decision thresholds. A particularly appropriate family of cost functions for this task is based on the receiver operating characteristics (ROC) curve. These functions are inherently robust to class imbalance and some of them are specifically optimized for high decision thresholds. The ROC curve for a binary classification problem plots the true positive rate (TPR) as a function of the false positive rate (FPR). Each point on the curve is obtained by choosing a classification threshold. The area under the ROC curve (AUC) is a measure of classifier performance.

1.1 Reasons to use ROC-based cost functions in drug discovery

In the following we outline five important reasons to use ROC-based cost functions (i.e. to optimize the AUC statistic or similar statistics relatively directly) in virtual screening, instead of using the typical cross-entropy cost function.

Class imbalance Typical cost functions such as cross-entropy do not work well when applied to datasets where class sizes are strongly imbalanced and classes are not easily separable. The minority class has little contribution to the cross-entropy cost function, making it inexpensive for the classifier to misclassify most of the minority class. In contrast, AUC-based cost functions sum over positive-negative sample pairs (rather than over individual samples), resulting in every class appearing equally often in the cost term. This makes AUC-based cost functions immune to class imbalance.

High decision thresholds Due to the high cost of *in vitro* experiments, only a small percentage of samples can be selected, i.e. the decision threshold must be high. The left part of the ROC corresponds to such high decision thresholds. Hence, *optimizing* this part of the ROC specifically targets our goals. This is why advanced quality metrics for virtual screening focus on the left part of the ROC [Mysinger and Shoichet, 2010]. In Section 2.2.2, we introduce a novel ROC-based cost function that focuses on maximizing the area under the left part of the ROC curve.

Not knowing the exact threshold in advance In virtual screening we may not know the decision threshold for classification in advance. If this is the case, it is reasonable to consider all realistic thresholds when measuring or optimizing classifier performance. The ROC (and AUC) consolidate classifier quality over all possible thresholds. The left part of the ROC (and quality metrics that focus on it) consolidate classifier quality across various high decision thresholds.

Better benchmarking results Evaluation metrics typically reported for methods benchmarking are based on AUC because of the reasons listed above. Thus, if many benchmarks use ROC-based quality metrics and one wants to surpass current methods in these benchmarks, one can ideally use cost functions that directly aim to optimize those quality metrics.

PU learning The machine learning task where only positive and unlabeled samples are available for training is referred to as *positive/unlabeled learning* (PU learning). An example application of PU learning in drug discovery is the classification of small molecules into drug-like and non-drug-like compounds. While we can assemble lists of drugs and known drug-like compounds as positive training samples, it is difficult to come up with lists of definitely non-drug-like compounds – or at least not any that are not trivially non-drug-like. However, we can come up with sets of unlabeled compounds with unknown drug likeness. Zhang and Lee [2008] show that PU learning is equivalent to treating the unlabeled samples as negative while optimizing an AUC-based cost function. Ren et al. [2018] show the effectiveness of AUC maximization for highly imbalanced PU learning for the special case where a part of the training data is mislabeled and/or some features are redundant.

1.2 Difficulties of AUC optimization and related work

While there are important reasons to optimize the AUC, it also gives rise to some challenging problems. Firstly, the gradient of AUC with respect to network weights is zero almost everywhere, because AUC changes only when the ranks of predictions for a positive sample and a negative sample are swapped. Therefore, AUC cannot be optimized directly by first-order optimization methods. Secondly, the AUC is a sum over pairs of samples rather than over individual samples, making its direct computation slow. In the following we outline typical solutions to these problems.

Approximations to AUC with non-zero gradient Different approximations to the AUC have been proposed that have non-zero gradients (on more than only a null set of weight space), allowing gradient-based optimization.

A sum of sigmoidal functions with the arguments scaled by a pre-defined value is a good approximation to the AUC [Yan et al., 2003]. However, it comes at the cost of creating very steep gradients depending on the choice of this scaling value, making optimization difficult. One of the differentiable

AUC approximations introduced by Yan et al. [2003] does not have the issue of steep gradients. The proposed objective function dynamically adjusts a sample pairs' contribution to the loss, based on the score difference between the positive and the negative sample. More specifically, if the difference is larger than a specified margin, then the contribution of a given pair to the loss is zero, otherwise it is a positive value that changes smoothly with the magnitude of the score difference. This formulation makes the optimization focus on maximizing the number of pairs that have a pairwise difference larger than a given margin, enhancing generalization performance. For these reasons, we base our proposed objectives on this loss function (named R_1) by Yan et al. [2003].

RankOpt [Herschtal and Raskutti, 2004] is a linear classifier which uses sigmoidal functions in the cost function with the scaling value being calculated from the data instead of fixing it a priori. The algorithm is also computationally efficient, making it linear in the number of samples as opposed to quadratic run time of other methods. However, being a linear classifier, RankOpt is not directly compatible with deep learning.

The AUC ignores the exact prediction scores (which might contain valuable additional information about the model's quality, for example scoring a positive as "0.9 (rank 1)" might indicate a more promising model than scoring it as "0.7 (rank 1)"). The AUC takes only ranks into account. (This limitation of the AUC also causes the zero-gradient problem.) To overcome this issue, a method called *scored ROC* was proposed [Wu et al., 2007] which is based on reducing scores for positives by a number between 0 and 1, the so-called margin. To this end, the *scored ROC curve* (not similar to ROC) plots margins against the corresponding AUC. The area under the scored ROC curve, called *scored AUC*, measures how quickly AUC declines when classifier outputs for positives are reduced. This metric has non-zero gradients with respect to sample scores.

Calders and Jaroszewicz [2007] use a polynomial (rather than sigmoidal) approximation of the step function. Their approximation allows to reformulate the sum over pairs of samples into a sum over individual samples, thus reducing the runtime of an epoch from quadratic to linear complexity. We achieve such runtime reduction with a sigmoidal approximation by using a lookup table that maps decision thresholds to false positive rates (see Section 2.2.2). Ferri et al. [2005] use a linear (rather than sigmoidal or polynomial) approximation of the step function. A study of the bias of several AUC approximations was performed by Vanderlooy and Hüllermeier [2008].

Direct optimization of AUC without gradient-based methods Optimizing AUC directly using coordinate descent (i.e. optimizing one model parameter at a time, which is feasible despite the zero gradient) yields good results for certain machine learning methods that were designed specifically for genetics applications [Zhu et al., 2017]. LeDell et al. [2016] introduce an ensemble approach based on *Super Learner* [van der Laan et al., 2007] which adjusts the combination of scores from several individual classifiers in favor of a higher AUC. They show that even though none of the base classifiers is specifically trained to maximize AUC, the Super Learner ensemble outperforms the top base algorithm, especially on data with high class imbalance.

Online AUC optimization We use out-of-batch predictions (see Section 2.2.1) and a lookup table (see Section 2.2.2). These techniques are related to online AUC optimization (i.e. training where data are not available all at once), which rewrites the sum of losses over sample pairs into a sum of losses of individual samples and uses buffers for positive and negative training samples [Zhao et al., 2011] or stores the first- and second-order statistics of training data [Gao et al., 2013].

2 Methods

In this section we will first introduce the AUC-based cost functions upon which our work relies, then we introduce the novel cost functions. The main contributions are: identification of five reasons for using ROC-based cost functions in virtual screening (Section 1.1), a novel algorithm *AUC-prev* that uses out-of-batch predictions for overall prediction improvement in AUC optimization (Section 2.2.1), and a new cost function $\mathcal{L}_{\text{logAUC}}$ that optimizes the ROC curve using a novel reweighting scheme for different decision thresholds, along with a lookup table for fast computation and a stop-gradient operator that prevents degenerate solutions (Section 2.2.2), and a comparison of five cost functions using four quality metrics and nine representative drug discovery datasets.

2.1 Approximation of the AUC

The AUC is the ratio of all positive-negative pairs where the positive has a higher prediction than the negative. In other words,

$$\text{AUC} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} H(x_i - y_j)}{mn}, \tag{1}$$

where $H(z) = \mathbb{1}[z > 0]$ is the Heaviside step function, the x_i are all m positive samples and the y_j are all n negative samples.

Yan et al. [2003] proposed an approximation to $H(x_i - y_j)$ with partially non-zero gradients:

$$f(x_i, y_j) = \begin{cases} (-(x_i - y_j - \gamma))^p, & \text{if } x_i - y_j < \gamma \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where $0 < \gamma \leq 1$ (usually $0.1 \leq \gamma \leq 0.7$) and $p > 1$ (usually 2 or 3) are hyperparameters. When comparing two classifiers that have the same AUC value, one of them may be better in the sense that it separates the positive scores from the negative scores by a larger margin. Equation (2) incorporates the margin γ to distinguish between these cases. The average of $f(x_i, y_j)$ taken over all pairs (x_i, y_j), i.e.

$$\mathcal{L}_{\text{AUC}} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} f(x_i, y_j), \tag{3}$$

is a cost function that encourages positive samples to have a score that is higher by at least γ than the score for negative samples. If this condition is not met, then that particular positive-negative pair contributes to the loss.

2.1.1 Lower-left part ROC curve optimization

The *left* part of the ROC changes if positive-negative pairs that have relatively *high* scores swap ranks. This portion of the ROC curve describes the classifier performance at high decision thresholds, i.e. when selecting only the top few percent of candidates, which is usually the case in drug discovery due to the high cost of subsequent *in vitro* experiments. To optimize for such situations, Yan et al. [2003] further modify Eq. (3) to transform the scores of positive and negative samples by the function $g(\cdot)$ before passing the pair to the cost function:

$$\mathcal{L}_{\text{leftAUC}} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} f(g(x_i), g(y_j)), \tag{4}$$

$$\text{where } g(s) = \begin{cases} (s - \beta\mu_s)^\alpha, & \text{if } s > \beta\mu_s \\ 0, & \text{otherwise} \end{cases}, \tag{5}$$

with hyperparameters $\alpha > 1$ but close to 1 (usually 1.1) and $\beta \geq 1$ (usually 1), and μ_s is the mean value of the classifier scores for all samples. Positive samples that have a high score will be mapped to a value that depends on the magnitude of the difference between the score of that sample and the classifier mean score. This has an effect of pushing these samples even more in the direction of high positive classification.

2.2 New cost functions

Based on the AUC and its approximation, Eq. (3), we propose two new objective functions for training any parametric classifiers to directly optimize the ROC curve using gradient-based methods. We use them with a multilayer perceptron with softmax outputs for applications in drug discovery.

2.2.1 AUC optimization using out-of-batch predictions

A mini-batch that consists of 0.1% of all positive and negative[1] samples computes 0.1% of the overall cost for usual cost functions, but only 0.0001% (if pairs are coherent, see Section 2.3) or less for

[1] The datasets we use have very few negative samples (not only in terms of relative class imbalance, but also in terms of absolute numbers), i.e. the advantages of the following method can be expected to be even more pronounced on larger (including imbalanced) datasets.

cost functions that are a sum over pairs of samples such as ROC-based ones. Thus, one “epoch” (i.e. seeing the overall loss once) requires an enormous number of mini-batches (quadratic rather than linear in the number of samples), and quick convergence to a good solution can be problematic. To address this problem, we propose using out-of-batch predictions for overall prediction improvement. In each iteration, the newest predictions for samples from the current mini-batch are used to update the weights of the network accordingly, while the *recent* predictions for all samples (including out-of-batch ones) are also used for loss computation, but considered constants that do not depend on the network weights. The proposed mini-batch-wise objective function looks as follows:

$$\mathcal{L}_{\text{AUC-prev}} = \mathcal{L}_{\text{AUC}}(\mathcal{X}_{\text{curr}}, \mathcal{Y}_{\text{curr}}) + \mathcal{L}_{\text{AUC}}(\mathcal{X}_{\text{curr}}, \mathcal{Y}_{\text{prev}}) + \mathcal{L}_{\text{AUC}}(\mathcal{X}_{\text{prev}}, \mathcal{Y}_{\text{curr}}), \tag{6}$$

where $\mathcal{X}_{\text{curr}}$, $\mathcal{Y}_{\text{curr}}$ are current predictions for positive resp. negative samples from the current mini-batch and $\mathcal{X}_{\text{prev}}$, $\mathcal{Y}_{\text{prev}}$ are the most recent predictions for *all* positive resp. negative samples. With this cost function, the network weights are optimized by not only considering the loss contribution of positive-negative pairs from the current mini-batch but also from pairs which consist of a current-mini-batch sample and an out-of-batch sample. Compared to other approaches, this procedure allows the loss to be based upon more samples than present in the mini-batch, reducing noise in the loss and making the training more stable.

2.2.2 Optimizing the area under the lin-log ROC curve

If the goal is to optimize for early enrichment, i.e. not missing out on good candidates at high decision thresholds, then a suitable performance measure is the area under a part of the lin-log ROC curve, called *logAUC* [Mysinger and Shoichet, 2010]. This quality measure shares some properties with the AUC statistic (e.g. robustness to class imbalance) but is biased towards early enrichment. The logAUC metric is often used for measuring the quality of methods that were trained with other metrics such as cross-entropy. Defining quality differently during training than during evaluation is suboptimal. We develop a new cost function targeted at directly optimizing the logAUC metric.

We observe that the AUC, Eq. (1), can be interpreted as integration over the ROC curve, which consists of n stripes of equal width:

$$\begin{aligned} \text{AUC} &= \frac{1}{mn} \sum_{j=1}^{n} \sum_{i=1}^{m} H(x_i - y_j) && (7) \\ &= \sum_{j=1}^{n} \left[\frac{j+1}{n} - \frac{j}{n} \right] \frac{1}{m} \sum_{i=1}^{m} H(x_i - y_j), && (8) \end{aligned}$$

where we assume the negatives to be sorted in order of descending classifier output. The term $\frac{1}{m}\sum_{i=1}^{m} H(x_i - y_j)$ corresponds to the TPR at the decision threshold y_j, i.e. to the height of the j^{th} stripe, and $\left[\frac{j+1}{n} - \frac{j}{n}\right] = \frac{1}{n}$ is the stripe width ($\frac{j}{n}$ and $\frac{j+1}{n}$ are the left and right coordinates of the stripe, respectively). So $\left[\frac{j+1}{n} - \frac{j}{n}\right] \frac{1}{m}\sum_{i=1}^{m} H(x_i - y_j)$ is the area of the j^{th} stripe, and the outer sum goes over all n stripes.

Computing the area under the part of the curve where FPR is in $[\lambda; 1]$ (with e.g. $\lambda = 0.001$) instead of $[0; 1]$ can be done as follows:

$$\text{AUC}_{\lambda} = \sum_{j=1}^{n} \left[\text{clip}_{[\lambda;1]}\left(\frac{j+1}{n}\right) - \text{clip}_{[\lambda;1]}\left(\frac{j}{n}\right) \right] \frac{1}{m} \sum_{i=1}^{m} H(x_i - y_j), \tag{9}$$

where $\text{clip}_{[a;b]}(z) = \min\{\max\{z, a\}, b\}$ clips the abscissa coordinates to the interval $[\lambda; 1]$.

The area logAUC under the log-transformed ROC curve can be computed by transforming the abscissa coordinates (in square brackets) to logarithmic scale:

$$\text{logAUC}_{\lambda} = \sum_{j=1}^{n} \left[\log\left(\text{clip}_{[\lambda;1]}\left(\frac{j+1}{n}\right)\right) - \log\left(\text{clip}_{[\lambda;1]}\left(\frac{j}{n}\right)\right) \right] \frac{1}{m} \sum_{i=1}^{m} H(x_i - y_j). \tag{10}$$

Clipping to $[\lambda; 1]$ is essential because otherwise logAUC would be infinite.

Equivalently to the explanation for AUC above, logAUC has gradient zero almost everywhere. In order to optimize logAUC with gradient-based methods, we approximate it by replacing the step function H in Eq. (10) by a smooth function, as was done for AUC in Eqs. (1)–(3). Defining $w_j := \log\left(\text{clip}_{[\lambda;1]}\left(\frac{j+1}{n}\right)\right) - \log\left(\text{clip}_{[\lambda;1]}\left(\frac{j}{n}\right)\right)$, we obtain the logAUC objective function

$$\mathcal{L}_{\text{logAUC}} = \sum_{j=1}^{n}\left(w_j \sum_{i=1}^{m} f(x_i, y_j)\right) = \sum_{j=1}^{n}\sum_{i=1}^{m} w_j\, f(x_i, y_j), \tag{11}$$

where the weighting factor w_j from Eq. (10) was pulled into the inner sum such that each positive-negative pair has a weighting factor for batch-based training.

This cost function directly optimizes the logAUC metric by individually scaling each of the n equal-width stripes which the AUC is composed of (like the logAUC metric does by log-transforming the abscissa of the ROC), thus giving more importance to the left part of the ROC curve.

Acceleration of rank computation There are a couple of implementation details worth mentioning. First, the usage of the weighting factor w_j requires computing the rank j of each negative sample (see the assumption under Eq. (8)) from its prediction y_j, which in turn requires sorting all negatives by their predictions. To save time, we construct a lookup table which maps thresholds to FPRs, allowing us to then estimate a negative's rank j. The lookup table is updated after every epoch (pass through all pairs) and uses equidistant keypoints in threshold space. Lookup uses linear interpolation. We found a step size of 0.001 to be a good trade-off between accuracy and time requirements.

This imprecise rank computation leads to imprecise cost gradients and degenerate solutions. Details and a remedy are described in the following. The score of each sample in a mini-batch is between two adjacent entries in the lookup table and we interpolate linearly between these two thresholds. Smaller predictions for *negatives* lead (by changing the interpolation weights) to higher estimates of sample ranks (because the prediction-to-rank mapping is a monotonically decreasing mapping). This causes the estimates of the sample weights w_j (estimated stripe widths in log space) to decrease. Decreasing estimates of w_j lead to a decreasing estimate of the overall loss, cf. Eq. (11). As a smaller loss is preferred by the optimization algorithm, and gradients can flow through this entire pipeline , predicted scores for negatives are incentivized to become smaller and converge to zero very quickly. Also the scores for positives converge to zero, apparently as a side effect, because the network does not have sufficient time/incentive to learn to distinguish them and simply learns to always output zero. The remedy is during an update step to conceal the strictly monotonous dependence of *rank estimates* on predictions, mimicking the zero gradient of the piecewise constant dependence of *actual ranks* on predictions. To this end, we use the stop-gradient operator $\text{sg}[\cdot]$ which sets gradients to zero when back-propagating through it. We replace w_j by $\text{sg}[w_j]$. This prevents degenerate solutions.

2.3 Experimental setup

Datasets Results are reported on nine large Quantitative Structure–Activity Relationship (QSAR) benchmark datasets [Butkiewicz et al., 2013] comprising small molecules labeled as active or inactive for nine protein targets. Each dataset was derived from a single screening effort in the PubChem database, and label accuracy for actives was confirmed via validating screens. As a cohesive set, these datasets avoid the construction biases often seen in other datasets, which can result in trivial decision boundaries and poor generalization (something which also affects [Chen et al., 2019] typical docking datasets such as DUD-E). They also display the realistically large numbers of diverse compounds and the class imbalances (a few hundred active molecules and inactives on the order of 10^5) typically seen in practical drug development projects, properties often lacking in model systems. See Butkiewicz et al. [2013] for details. As input features, we used the 391 descriptors from the Reduced Short Range descriptor set, which has previously been identified as sufficiently informative and compact [Mendenhall and Meiler, 2016, Vu et al., 2019]. We normalized features using z-score scaling, which was found to be the most effective for these datasets [Mendenhall and Meiler, 2016].

Network architecture For our experiments we adopt many of the architecture design choices that have proven to be useful [Mendenhall and Meiler, 2016] on the same PubChem QSAR datasets with standard cost functions. Throughout the experiments we use a two-layer feed-forward neural network with one hidden layer of thirty-two neurons with rectified linear units as activation functions and one output unit with sigmoidal activation function. Also, we use dropout in the input layer and in the first hidden layer. We use dropout rates that were optimized by Mendenhall and Meiler [2016] for each of

Table 1: Evaluation using "AUC" as a quality metric. The loss functions $\mathcal{L}_{\text{AUC}}$ and $\mathcal{L}_{\text{AUC-prev}}$ proposed for optimizing this quality metric are highlighted in black, other loss functions in grey. Results that significantly ($p < 0.05$) outperform the baseline method $\mathcal{L}_{\text{cross-entropy}}$ are marked **bold**. On all 4 datasets on which $\mathcal{L}_{\text{AUC-prev}}$ outperforms $\mathcal{L}_{\text{cross-entropy}}$, it also outperforms $\mathcal{L}_{\text{AUC}}$, indicating advantages of our proposed usage of out-of-batch samples over usual batch-wise training. Error margins range from ± 0.0004 to ± 0.0019.

SAID	$\mathcal{L}_{\text{cross-entropy}}$	$\mathcal{L}_{\text{AUC}}$	$\mathcal{L}_{\text{AUC-prev}}$	$\mathcal{L}_{\text{leftAUC}}$	$\mathcal{L}_{\text{logAUC}}$
1798	0.727	**0.767**	**0.769**	**0.759**	**0.752**
1834	0.939	**0.944**	0.934	0.936	**0.951**
2258	0.809	**0.819**	**0.826**	0.810	0.773
2689	0.870	**0.877**	**0.878**	0.859	0.865
435008	0.785	0.778	0.770	0.772	0.775
435034	0.814	**0.816**	0.808	**0.818**	**0.832**
463087	0.861	0.860	0.859	0.846	0.862
485290	0.753	0.756	0.748	0.748	**0.760**
488997	0.767	0.769	**0.773**	0.762	0.763

the nine datasets individually. Using dropout has been proven to be much more beneficial for QSAR datasets than employing additional hidden layers or larger neural networks, whose effect is shown to be insignificant in comparison [Mendenhall and Meiler, 2016].

Training procedure We train on mini-batches which are constructed as follows: each mini-batch contains coherent positive-negative pairs, i.e. we randomly uniformly draw positive and negative samples from the training set and construct all possible pairs between these. The use of coherent mini-batches has many advantages such as being memory-efficient, easy to implement, and parallelizable.

All parameters are initialized using the scheme by He et al. [2015]. For training, we used the Adam optimization algorithm [Kingma and Ba, 2014] with different learning rates for each task, exponential decay rate for the first and second moment at 0.9 and 0.999, respectively. The optimal learning rates were found by K-fold cross-validation and testing approach as in [Korjus et al., 2016] with $K = 4$. The best learning rate was 0.001 for all nine tasks and all objective functions, except for $\mathcal{L}_{\text{logAUC}}$ on dataset ID 2258, where the best learning rate was 0.003.

3 Results and discussion

Similar to [Mysinger and Shoichet, 2010], model performance is evaluated by computing logAUC for FPR in $[0.001; 0.1]$. This quality metric is very popular for chemistry-related problems such as drug discovery, where we focus on high decision thresholds. In addition to this quality metric we report two other metrics that focus on high decision thresholds, namely AUC for FPR in $[0.001; 0.1]$ and logAUC for FPR in $[0.001; 1]$; as well as AUC.

Tables 1–4 report these four quality metrics for each of the four ROC-based objective functions. Additionally, results are compared to the baseline method, i.e. the cross-entropy objective. Each row represents one of the nine datasets. Bold faced numbers indicate that the corresponding objective function significantly ($p < 0.05$) outperforms the baseline for that particular dataset. Confidence intervals for each metric were computed by bootstrapping the test set with replacement 200 times [Mendenhall and Meiler, 2016]. For each metric, the results of the cost functions proposed specifically for that metric are shown in black. The results of other cost functions are also shown, but greyed out.

Table 1 shows results using the AUC quality metric. The cost functions designed specifically to optimize this metric are $\mathcal{L}_{\text{AUC}}$ and $\mathcal{L}_{\text{AUC-prev}}$. The cost function $\mathcal{L}_{\text{AUC}}$ outperforms the baseline method $\mathcal{L}_{\text{cross-entropy}}$ at a significance level of $\alpha = 0.05$ on 5 out of the 9 datasets. Our procedure $\mathcal{L}_{\text{AUC-prev}}$ of using out-of-batch predictions outperforms the baseline method significantly ($\alpha = 0.05$) on 4 out of 9 datasets, including one dataset on which $\mathcal{L}_{\text{AUC}}$ does not outperform $\mathcal{L}_{\text{cross-entropy}}$. Moreover, on all

Table 2: Evaluation using "AUC for FPR in $[0.001; 0.1]$" as a quality metric. The loss function $\mathcal{L}_{\text{leftAUC}}$ designed to optimize this quality metric is highlighted in black. It significantly ($p < 0.05$, **bold**) outperforms $\mathcal{L}_{\text{cross-entropy}}$ on 4 out of 9 datasets. $\mathcal{L}_{\text{logAUC}}$, designed for a similar purpose, significantly outperforms $\mathcal{L}_{\text{cross-entropy}}$ on 5 out of 9 datasets. Error margins range from ± 0.0011 to ± 0.0028.

SAID	$\mathcal{L}_{\text{cross-entropy}}$	$\mathcal{L}_{\text{AUC}}$	$\mathcal{L}_{\text{AUC-prev}}$	$\mathcal{L}_{\text{leftAUC}}$	$\mathcal{L}_{\text{logAUC}}$
1798	0.272	**0.315**	0.260	0.247	0.211
1834	0.622	**0.653**	0.606	0.591	**0.662**
2258	0.523	**0.547**	0.489	0.511	0.298
2689	0.589	**0.621**	**0.603**	**0.601**	0.589
435008	0.370	**0.376**	0.361	**0.389**	0.364
435034	0.388	0.388	0.386	0.380	**0.401**
463087	0.342	0.338	0.339	0.318	**0.362**
485290	0.392	0.393	0.385	**0.403**	**0.418**
488997	0.383	**0.451**	**0.444**	**0.453**	**0.432**

Table 3: Evaluation using "logAUC for FPR in $[0.001; 1]$" as a quality metric. The loss function $\mathcal{L}_{\text{logAUC}}$ designed to optimize this quality metric significantly ($p < 0.05$, **bold**) outperforms $\mathcal{L}_{\text{cross-entropy}}$ on 7 out of 9 datasets. Error margins range from ± 0.0007 to ± 0.0022.

SAID	$\mathcal{L}_{\text{cross-entropy}}$	$\mathcal{L}_{\text{AUC}}$	$\mathcal{L}_{\text{AUC-prev}}$	$\mathcal{L}_{\text{leftAUC}}$	$\mathcal{L}_{\text{logAUC}}$
1798	0.329	**0.353**	**0.333**	0.324	0.298
1834	0.575	**0.589**	0.557	0.562	**0.629**
2258	0.489	**0.514**	0.476	0.482	0.411
2689	0.531	**0.545**	**0.539**	**0.538**	**0.558**
435008	0.391	**0.396**	0.390	**0.403**	**0.399**
435034	0.415	0.415	0.411	0.415	**0.430**
463087	0.397	0.398	0.393	0.385	**0.404**
485290	0.418	**0.425**	0.411	0.419	**0.427**
488997	0.395	**0.425**	**0.417**	**0.425**	**0.440**

4 datasets on which $\mathcal{L}_{\text{AUC-prev}}$ outperforms $\mathcal{L}_{\text{cross-entropy}}$, it also outperforms $\mathcal{L}_{\text{AUC}}$ (with $p < 0.05$ on two of the datasets), indicating advantages of our proposed usage of out-of-batch samples.

Results in Table 2 are evaluated using the "AUC for FPR in $[0.001; 0.1]$" quality metric. The function $\mathcal{L}_{\text{leftAUC}}$ was designed specifically for optimizing this metric. The results show that $\mathcal{L}_{\text{leftAUC}}$ outperforms the cross-entropy baseline in 4 out of 9 datasets at a significance level of $\alpha = 0.05$. Our objective function $\mathcal{L}_{\text{logAUC}}$ (designed for a similar purpose) significantly outperforms $\mathcal{L}_{\text{cross-entropy}}$ in 5 out of 9 datasets. This indicates that an ROC-based cost function that maximizes the area under the left part of the ROC can improve classifier performance as compared to typical cost functions (such as cross-entropy) when the goal is to optimize the performance at high decision thresholds. In addition, $\mathcal{L}_{\text{logAUC}}$ significantly outperforms $\mathcal{L}_{\text{leftAUC}}$ in 4 out of 9 datasets.

Table 3 demonstrates results for the "logAUC for FPR in $[0.001; 1]$" quality metric. Our proposed objective function $\mathcal{L}_{\text{logAUC}}$ was specifically designed for this metric and performs significantly better ($\alpha = 0.05$) than cross-entropy on 7 out of 9 datasets. The results also show that $\mathcal{L}_{\text{logAUC}}$ outperforms other ROC-based objective functions on many of the datasets. Specifically, it significantly outperforms $\mathcal{L}_{\text{AUC}}$ on 5 out of 9 datasets.

Table 4: Evaluation using "logAUC for FPR in $[0.001; 0.1]$" as a quality metric. The loss function $\mathcal{L}_{\text{logAUC}}$ designed to optimize this quality metric significantly ($p < 0.05$, **bold**) outperforms $\mathcal{L}_{\text{cross-entropy}}$ on 7 out of 9 datasets. Error margins range from ± 0.0008 to ± 0.0025.

SAID	$\mathcal{L}_{\text{cross-entropy}}$	$\mathcal{L}_{\text{AUC}}$	$\mathcal{L}_{\text{AUC-prev}}$	$\mathcal{L}_{\text{leftAUC}}$	$\mathcal{L}_{\text{logAUC}}$
1798	0.154	**0.164**	0.143	0.141	0.101
1834	0.390	**0.408**	0.367	0.372	**0.466**
2258	0.339	**0.372**	0.306	0.330	0.238
2689	0.367	**0.382**	**0.373**	**0.384**	**0.413**
435008	0.219	**0.229**	**0.224**	**0.241**	**0.231**
435034	0.229	0.227	0.228	0.226	**0.245**
463087	0.176	0.175	0.171	0.171	**0.189**
485290	0.268	**0.279**	0.265	**0.277**	**0.281**
488997	0.228	**0.268**	**0.259**	**0.275**	**0.297**

Table 4 shows results for the "logAUC for FPR in $[0.001; 0.1]$" quality metric which is also optimized by our $\mathcal{L}_{\text{logAUC}}$ objective. The results demonstrate that, again, our objective function outperforms the cross-entropy baseline in 7 out of 9 datasets and $\mathcal{L}_{\text{AUC}}$ in 5 out of 9 datasets, both at a significance level of $\alpha = 0.05$.

One important aspect to mention is that when using $\mathcal{L}_{\text{cross-entropy}}$ we oversample the minority class (positives) in order to compare the AUC-based cost functions against the cross-entropy under ideal circumstances. Thus we should expect even more favorable results compared with cross-entropy if no oversampling is performed. The AUC-based cost functions are robust towards class imbalance, i.e. perform equally well without oversampling. Thus, they do not require tuning the oversampling ratio, as the cross-entropy loss does. On the other hand, they have additional hyperparameters that require tuning. Luckily, a wide range of values works well in practice [Yan et al., 2003].

4 Conclusions

We listed a series of special properties of virtual screening datasets, such as class imbalance, all of which can be addressed by using ROC-based cost functions. Such cost functions, in turn, have peculiarities such as the necessity for techniques to avoid zero gradients (which we borrowed from literature), and a quadratic rather than linear number of summands (which we addressed by proposing to use out-of-batch samples and coherent mini-batches). Moreover, to optimize performance specifically for the high decision thresholds that are used in virtual screening, and to more directly optimize the logAUC quality metric that is popular in this domain, we proposed an approximation $\mathcal{L}_{\text{logAUC}}$ to logAUC with nonzero gradients. To accelerate its computation, we replaced precise computation by a lookup table, and to prevent the wrong gradient caused by this replacement from leading to degenerate solutions, we used a stop-gradient operator. Our methods outperformed cross-entropy in many scenarios in a benchmark of realistic diverse datasets. We do not claim that these AUC losses are perfect for all situations; rather, we exemplify how losses can be aligned with project-specific goals. We encourage active exploration of loss options in virtual screening and in other applications, instead of following the old tradition of resorting to cross-entropy "by default".

Acknowledgements

We thank Benjamin P. Brown for helpful discussions.

References

M. Butkiewicz, E. W. Lowe, R. Mueller, J. L. Mendenhall, P. L. Teixeira, C. D. Weaver, and J. Meiler. Benchmarking ligand-based virtual high-throughput screening with the PubChem database. *Molecules*, 18(1): 735–756, 2013.

T. Calders and S. Jaroszewicz. Efficient AUC optimization for classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53. Springer, 2007.

L. Chen, A. Cruz, S. Ramsey, C. J. Dickson, J. S. Duca, V. Hornak, D. R. Koes, and T. Kurtzman. Hidden bias in the DUD-E dataset leads to misleading performance of deep learning in structure-based virtual screening. *PloS one*, 14(8), 2019.

C. Ferri, P. Flach, J. Hernández-Orallo, and A. Senad. Modifying roc curves to incorporate predicted probabilities. In *Proceedings of the ICML 2005 workshop on ROC Analysis in Machine Learning*, 2005.

W. Gao, R. Jin, S. Zhu, and Z.-H. Zhou. One-pass AUC optimization. In *International Conference on Machine Learning*, pages 906–914, 2013.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

A. Herschtal and B. Raskutti. Optimising area under the ROC curve using gradient descent. In *Proceedings of the twenty-first international conference on Machine learning*, page 49. ACM, 2004.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

K. Korjus, M. N. Hebart, and R. Vicente. An efficient data partitioning to improve classification performance while keeping parameters interpretable. *PloS one*, 11(8):e0161788, 2016.

E. LeDell, M. J. van der Laan, and M. Peterson. Auc-maximizing ensembles through metalearning. *The international journal of biostatistics*, 12(1):203–218, 2016.

J. Mendenhall and J. Meiler. Improving quantitative structure–activity relationship models using artificial neural networks trained with dropout. *Journal of computer-aided molecular design*, 30(2):177–189, 2016.

M. M. Mysinger and B. K. Shoichet. Rapid context-dependent ligand desolvation in molecular docking. *Journal of chemical information and modeling*, 50(9):1561–1573, 2010.

K. Ren, H. Yang, Y. Zhao, M. Xue, H. Miao, S. Huang, and J. Liu. A robust AUC maximization framework with simultaneous outlier detection and feature selection for positive-unlabeled classification. *CoRR*, abs/1803.06604, 2018. URL `http://arxiv.org/abs/1803.06604`.

M. J. van der Laan, E. C. Polley, and A. E. Hubbard. Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), Jan. 2007. doi: 10.2202/1544-6115.1309. URL `https://doi.org/10.2202/1544-6115.1309`.

S. Vanderlooy and E. Hüllermeier. A critical analysis of variants of the auc. *Machine Learning*, 72(3):247–262, 2008.

O. Vu, J. Mendenhall, D. Altarawy, and J. Meiler. Bcl:: Mol2d—a robust atom environment descriptor for qsar modeling and lead optimization. *Journal of computer-aided molecular design*, 33(5):477–486, 2019.

S. Wu, P. Flach, and C. Ferri. An improved model selection heuristic for auc. pages 478–489, 09 2007. doi: 10.1007/978-3-540-74958-5_44.

L. Yan, R. H. Dodier, M. Mozer, and R. H. Wolniewicz. Optimizing classifier performance via an approximation to the Wilcoxon–Mann–Whitney statistic. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 848–855, 2003.

D. Zhang and W. S. Lee. Learning classifiers without negative examples: A reduction approach. In *Digital Information Management, 2008. ICDIM 2008. Third International Conference on*, pages 638–643. IEEE, 2008.

P. Zhao, S. C. Hoi, R. Jin, and T. Yang. Online AUC maximization. 2011.

L. Zhu, H.-B. Zhang, and D.-S. Huang. Direct AUC optimization of regulatory motifs. *Bioinformatics*, 33(14): i243–i251, 2017.